December 2002

The Center for INFOSEC Studies and Research

white paper

An Editor for Adaptive XML-Based Policy Management of IPsec

Raj Mohan, Timothy E. Levin, Cynthia E. Irvine

Center for Information Systems Security Studies and Research
Computer Science Department
Naval Postgraduate School
Monterey, California 93943

An Editor for Adaptive XML-Based Policy Management of IPsec

Raj Mohan
Indian Army
rmohan@nps.navy.mil

Timothy E. Levin
Naval Postgraduate School
televin@nps.navy.mil

Cynthia E. Irvine
Naval Postgraduate School
irvine@nps.navy.mil

Abstract

TCP/IP provided the communications foundation for the Internet and the IPsec protocol now promises to enable the desired security strength. IPsec provides users with a mechanism to enforce a range of security services for both confidentiality and integrity, enabling them to securely pass information across networks. Dynamic parameterization of IPsec further enables security mechanisms to adjust the level of security service "on-the-fly" to respond to changing network and operational conditions. The IPsec implementation in OpenBSD works in conjunction with the Trust Management System, KeyNote, to achieve this. However the KeyNote engine requires that an IPsec policy be defined in the KeyNote specification syntax. Defining such a dynamic security policy in the KeyNote Policy Specification language is, however, complicated and could lead to incorrect specification of the desired policy, thus degrading the security of the network. We present an alternative XML representation of this language and a graphical user interface to create and manage a consistent and correct security policy. The interface has the simplicity of a simple menu-driven editor that not only provides KeyNote with a policy in the specified syntax but also integrates techniques for correctness verification and validation.

1 Introduction

1.1 Objective

Network Protocols such as IPsec and trust management systems like Keynote provide mechanisms to secure computer-to-computer communications. These tools enable the user to use various encryption and authentication mechanisms to ensure confidentiality, integrity and non-repudiation of communications. Dynamic parameterization [1] of IPsec further enables security mechanisms to adjust the level of security service "on-the-fly" to respond to changing network and operational conditions. The trust management system, Keynote specifies a language for describing actions, which are operations with security consequences that are to be controlled by the system [3][4][5]. The language also provides the syntax for specifying the application policies, which govern the actions that the *principals* are authorized to perform. To translate a dynamic organizational security policy into the Keynote specification language is however a daunting task due to the complexities of the language and the policy. An incorrect specification of the security policy might result in a compromise of network security. It is in this context that the need for an alternative policy specification mechanism is perceived. This mechanism should enable the user to clearly and correctly specify the policy and also verify that the specified policy is free of inconsistencies and contradictions. The purpose of this work is to analyze, design and implement a policy editor interface that guides a user to specify various attributes of the IPsec security policy. The policy is stored in an intermediate XML format. The program will automatically generate the equivalent policy in the Keynote specification language. A presentation mechanism will be described for providing the user with an intuitive presentation that can be helpful in preventing inconsistencies and contradictions in the specified policy.

1.2 Background

The increased dependence on computers for communication has enhanced the importance of network security. The use of the inherently insecure Internet as the medium for communicating sensitive material requires that the end users have capabilities to ensure that the data transmitted is secure. Furthermore, network administrators should have means to translate the desired organizational security policy into an automated security policy and have mechanisms to implement this policy over their network.

IPsec extends the IP Protocol to enable security for TCP/IP communications. IPsec provides both secrecy and integrity services. A wide variety of choices are available when establishing protected communications across the network. The appropriate choice and combination of secrecy and integrity mechanisms will depend upon the "trust relationships" between the communicating entities. Those relationships are constrained by the policy of each entity. Negotiation of policy and mechanisms takes place in the context of the Internet Key Exchange (IKE) framework and the Internet Security Association and Key Management Protocol (ISAKMP) [13]. However, IKE and ISAKMPD do not provide a general mechanism for managing and incorporating security policy. In order to ensure that IPSEC consistently meets the local security policy needs of the user, a Trust Management System may be used to encode policy and support communications security negotiation and management [17].

A trust management system unifies the elements of security policy, credentials, access control, and authorization. The IPsec implementation in OpenBSD utilizes a trust management system to manage security according to policy [2]. Applications can use the Keynote trust management system to verify, through the compliance checker, whether a requested policy addition or change is authorized [6].

Quality of Security Service (QoSS) provides a means to manage security services based on the requirements set by the user's requests, the system's security policy, the availability of system resources and the network environment [9]

Dynamic parameterization of IPsec [1] provides more granularity in IPsec and provides flexibility to adjust security controls according to changes in threat conditions, critical time transmissions, and network congestion/traffic. This makes IPsec a QoSS mechanism.

All the above mechanisms depend on having in place a correct security policy specified in the Keynote specification language. For any practical real life network operations specifying such a dynamic and granular policy is an almost insurmountable task due to the syntactic complexity of the KeyNote language and the inherent complexity of the policy logic involved. An XML-based specification of the policy should provide the desired flexibility, be easy to use and support an interface for administration of the security policy. This would provide an abstraction to the KeyNote language and enable users to better utilize the power of IPsec and KeyNote in managing network security [14].

1.3 Expected Benefits

By providing a policy management toolkit it will be possible to unleash the power of IPsec usage and will enable government and military security systems to automate security service adjustments according to dynamic environmental parameter settings, such as INFOCON and THREATCON. The use of XML in such an effort will enable common use of available XML tools for ensuring security policy consistency and also utilize the flexibility and compatibility that XML provides. The power of XML security can also be harnessed to enhance the overall security of the communicating systems. An easy to use interface ensures its use and the resulting policy correctness will provide confidence in the overall security implementation of the network.

1.4 Organization of this Paper

The paper will be organized as follows: Section 2, Related Work, consists of a brief survey of related research. Section 3, Keynote Support for QoSS, reviews the Keynote language and its specification for the QoSS implementation in OpenBSD 2.8. Section 4, XML and Policy Representation, describes XML technologies and their application to the problem domain. Section 5, Design and Implementation, presents the design philosophy of the toolkit, the considerations and overall architecture will be discussed in detail. Implementation issues of the components will be highlighted in this section. Section 6 describes future work and Section 7 summarizes our results.

2 Related Work

2.1 IPsec

The popularity of the Transfer Control Protocol/ Internet Protocol (TCP/IP) and the growing use of computer networks for governmental and business made these protocols vulnerable to scrutiny, attacks and misuse. TCP/IP, which was designed to provide packet based communications over unreliable telephone networks, was not designed for providing secure communications. The first attempt to provide security involved a simple "protect-all" approach to network security i.e. the Virtual Private Networks (VPN). IPsec was then developed to address security vulnerabilities inherent in the Internet Protocol (IP), by defining a more flexible security mechanism for sending data across an insecure medium. IPsec introduced the ability to provide a range of security services ultimately defined by a security policy (See Figure 1). The security policy defines specific security services for each packet, according to packet characteristics such as source and destination addresses [11].

IPsec provides security services at the IP layer by enabling a system to select required security protocols, determine the algorithm(s) to use for the service(s), and put in place any cryptographic keys required to support the requested services. There are two extension headers that follow the main IP header and which incorporate the security features of IPsec. The extension header for authentication is known as the Authentication Header (AH) and that for encryption is known as the Encapsulating Security Payload (ESP) header. The difference between the AH and the authentication within ESP is essentially the amount of content of the packet and headers that is authenticated. Figure 2 and Figure 3 depict the coverage of ESP and AH respectively.

IPsec was designed to provide an efficient and effective cryptographic security mechanism for IP version 4 and IP version 6. The mechanism provides the following services: access control, connectionless integrity, data origin authentication, protection against replays,(a form of partial sequence integrity), confidentiality (encryption), and limited traffic flow confidentiality. These services are applied at the IP layer, providing security for IP and/or upper layer protocols [12][18]. The cryptographic algorithms are applied in accordance with system security policies that are defined within IPsec. IPsec can be used on a variety of system architecture models: host-to-host, gateway-to-gateway and gateway-to-host/host-to-gateway [8].

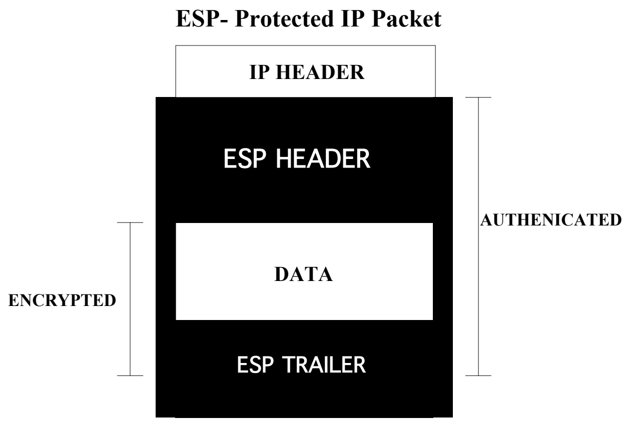


Figure 1. ESP- Protected IP Packet. (After: [8] p. 49)

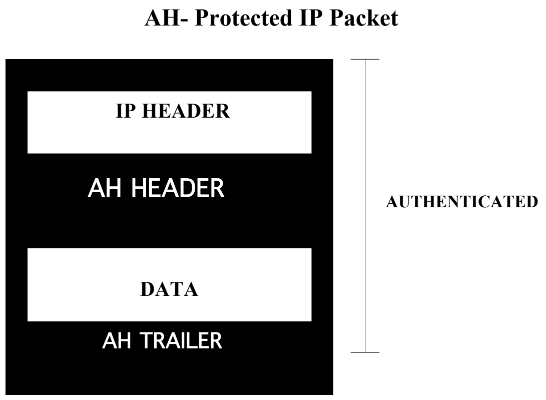


Figure 2. AH-Protected IP Packet. (After: [8], p. 51)

2.1.1 Security Associations. A key concept that appears in both the authentication and confidentiality mechanisms for IP is the Security Association (SA). An association is a one-way relationship between the sender and a receiver that affords security services to the traffic carried on it. If a peer relationship is needed for two-way secure exchange, then two security associations are required. A Security Association is uniquely defined by three parameters: Security Parameters Index (SPI), IP destination address, and Security protocol identifier.

2.1.2 SA Selectors. IPsec provides the user with considerable flexibility in the way in which IPsec services are applied to IP traffic. SAs can be combined in a number of ways to yield the desired security configuration. Furthermore, IPsec provides a high degree of granularity in discriminating between traffic that is afforded IPsec protection and traffic that is allowed to bypass IPsec, in the former case relating IP traffic to specific SAs.

The means by which IP traffic is related to specific SAs (or no SAs in the case of traffic allowed to bypass IPsec) is the nominal Security Policy Database (SPD). Each SPD entry is defined by a set of IP and upper-layer protocol field values, called selectors. In effect, these selectors are used to filter outgoing and incoming traffic in order to map it into a particular SA. Selectors are of many types for e.g. destination IP address, transport layer protocol, IP Sec protocol (AH or ESP or AH/ESP), source and destination ports etc.

2.2 Keynote trust management system

OpenBSD IPsec incorporates the concept of trust and security policy management by implementing KeyNote. The research utilizes the OpenBSD IPsec mechanism as a model for discussion and implementation. Figure 7 depicts the KeyNote trust management process.

KeyNote is a simple and flexible trust-management system designed to work well for a variety of large- and small- scale Internet-based applications. It provides a single, unified language for both local policies and credentials. KeyNote policies and credentials, called `assertions'; contain predicates that describe the trusted actions permitted by the holders of specific public keys. KeyNote assertions are essentially small, highly structured programs. A signed assertion, which can be sent over an un-trusted network, is also called a `credential assertion'. Credential assertions, which also serve the role of certificates, have the same syntax as policy assertions but are also signed by the principal delegating the trust.

In KeyNote:

- Actions are specified as a collection of name-value pairs. For instance a name value pair could be app_dom = ”email”. These are called as action attributes and a query is made with the action attributes and their associated values.
- Principal names can be any convenient string and can directly represent cryptographic public keys.
- The same language is used for both policies and credentials.
- The policy and credential language is concise, highly expressive, human readable, and compatible with a variety of storage and transmission media, including electronic mail.
- The compliance checker returns an application-configured `policy compliance value' that describes how a request should be handled by the application. Policy compliance values are always derived from policy and credentials, facilitating analysis of KeyNote-based systems.
- Compliance checking is efficient enough for high-performance and real-time applications.

Despite these advantages, the KeyNote Policy language has some technical challenges regarding understandability of complex policies, which are described in later sections.

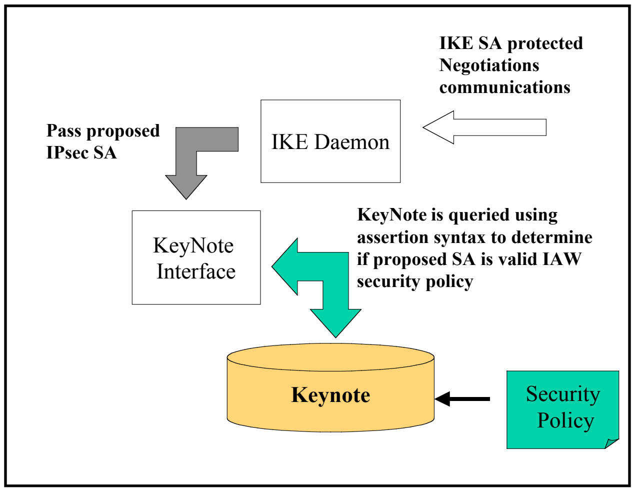

IKE SA protected Negotiations communications
Pass proposed IPsec SA
IKE Daemon
KeyNote Interface
KeyNote is queried using assertion syntax to determine if proposed SA is valid IAW security policy
Keynote
Security Policy

Figure 3. KeyNote Process (After [1])

2.3 Quality of Security Service (QOSS)

IPsec provides a high degree of granularity in discriminating between traffic that is afforded IPsec protection and traffic that is allowed to bypass IPsec. Further use of a trust management system such as Keynote enables an application to simply ask the compliance checker whether a requested action should be allowed. Thus if we specify a granular security policy as permissible by IPsec and use Keynote to verify a request based on the policy, we would be able to modulate the security settings of applications dynamically in accordance with the security and performance requirements of the applications in particular, and networks as a whole. This is the essence of 'Quality of Security Service' (QoSS).

Similar to the modulation of resources to support QoS, security services can be defined in terms of user and system requirements, network environment factors and available resources. Without a range of security services, a user is faced with the rigid and limited choice of "all or nothing" for each security service. Historically, security services have been provided in such a static manner [16]. Quality of Security Service (QoSS) provides a more flexible solution to the provision of security services. The security resource manager and/or the security system can adjust security service to meet user requirements, system security policy and network environment constraints [9].

QoSS has several mechanisms to handle security variablity. A security variance for a particular policy exists when that policy may be enforced utilizing a specific range of security attributes. Therefore, based on the policy parameters, the attributes used to enforce the security policy may differ according to selection criteria such as "network mode." Fixed requirements are used to set minimum level acceptable security standards. A range of security settings meeting or exceeding this minimum level can be provided. For example a system may utilize SHA as a minimum level authentication algorithm for all message handling. Users or applications could apply further granularity in support of confidentiality to messages by selecting an encryption algorithm from a provided range. Other examples of variable security attributes that may be used are: assurance level, key length or security attribute expiration date stamp [9].

2.3.1 Dynamic Parameters and Network Modes. Government and DOD organizations utilize a variety of dynamic parameters to define a predefined response of specific actions according to policy. Examples include INFOCON and THREATCON levels. In order for a security mechanism to be fully functional within the DOD and Government infrastructure, it has to be able to incorporate the dynamic parameters into the security setting decision-making process. A change in an INFOCON or THREATCON level should have an immediate effect on attributes and settings in a security mechanism. By introducing a dynamic mechanism, a system can modulate its security settings in response to these dynamic parameters. Security level and network mode, defined in the following sections, have been chosen as two abstract dynamic parameters that govern changes to security attributes as defined in the organization's security policy [15].

In the approach described here dynamic network parameters are represented as network modes. We use the following network modes: normal, impacted, and crisis. Normal mode is defined as ordinary operating conditions with normal traffic load and no heightened threat conditions. Impacted mode may be defined when the network/system is experiencing high levels of traffic and therefore certain security selection may not be available due to efficiency constraints. Emergency mode may be defined as a situation that requires the highest level of security or the lowest level dependent on the situation and policy[16].

2.3.2 User Choices for Security Levels. Security classification levels are a common metric used in the government and DOD to distinguish authorization for classified information. Common levels include Top Secret, Secret, Classified and Unclassified. Each of these levels correspond to different governing policies and requirements associated with the threat to national security by the disclosure of information to adversaries. Likewise, security selection levels, as defined here for proof of concept, represent an increasing requirement for stronger security (e.g. encryption and authentication algorithms).

Network security policies may utilize a range of maximum and minimum-security levels for each variant security service. Minimum-security levels set the lowest acceptable security attributes and maximum-security levels establish a ceiling on the use of available security resources. Intersections of policies require further granularity in security settings to satisfy all governing users and systems. A user may also desire to select a higher level of security than the predefined minimum [15].

A user or application, however, may quickly become overwhelmed with the security setting details, potentially resulting in degraded security or performance. By developing security definitions that encompass detailed security settings required by users or applications, the complexity of the selection process for the security settings can be simplified to a reasonable level. One approach would involve the use of the following Network Security levels: high, medium and low [16]. 'High' security level would utilize strong levels of security attributes, medium level, moderate level of security attributes, and low level, low to no security attributes.
By implementing this approach the system security resource manager or security engineer is responsible for presetting security variables and ranges in accordance with choices offered to users or applications. A mapping of allowable security settings to security levels, providing a range of selection or specific values, is required to properly enforce the system security policy [15].

2.3.3 Mapping Abstract Parameters to Security Mechanism. A mapping of abstract dynamic parameters to resident security mechanisms is required to properly enforce policy decisions. For example, network modes may be mapped to security level ranges and ultimately to security attributes and settings.

Table 1. Mapping Security Policies to Security Attributes. (From [1])

Network Mode	Security Level	Security Attributes
Normal	Low	Encryption: None Authentication: MD5
	Medium	Encryption: DES Authentication: MD5
	High	Encryption: 3DES Authentication: MD5
Crisis	Low	Encryption: None Authentication: None
	Medium	Encryption: None Authentication: None
	High	Encryption: DES Authentication: MD5
Impacted	Low	Encryption: 3DES Authentication: MD5
	Medium	Encryption: 3DES Authentication: SHA
	High	Encryption: AES Authentication: SHA

The security resource manager and security engineer would define the network modes and security levels to provide the users and applications with appropriate security service as translated into QoSS choices. Once defined, the complexity of the security mechanism and security attribute selection is transparent to the user. (See Figure 9)

2.4 Implementation Issues

Quality of Security Service (QoSS) provides us with a mechanism to modulate the security settings and enhance performance based on both necessity (e.g. threat) and resource availability. It also provides us with a tool to ensure that the minimum-security requirements of applications and the network as proposed in the security policy is not violated. Hence defining an adaptive security policy based on network threat and performance conditions is the key to optimal and secure utilization of the network resources. Keynote provides one such policy specification language but its practical implementation with complex policy statements extremely difficult. An abstraction for this language is therefore felt necessary. It would use the power of Keynote for formal compliance checking and at the same time be easy to use and administer. This is dealt with in the following chapters in detail.

3 Keynote Support for QoSS

The syntax and semantics of the Keynote language is described in detail in RFC 2704 [3]. In this section a brief overview of the language and the specific parts that need emphasis will be highlighted. The language is used for specifying application 'policies,' which govern the actions that principals (entities that can be authorized to perform actions) are authorized to perform. The language provides the semantics for describing 'actions,' which are operations with security consequences that are to be controlled by the system. It is also used for specifying 'credentials', which allow principals to delegate authorization to other principals.

Keynote assertions are divided into sections, called 'fields' that serve various semantic functions. Each field starts with an identifying label at the beginning of a line, followed by the ":" character and the fields contents. There can be at most one field per line.
One mandatory field is required in all assertions:

- Authorizer

Six optional fields may also appear:

- Comment
- Conditions
- KeyNote-Version
- Licensees
- Local-Constants
- Signature

The conditions field is used to define a security policy. This field gives the 'conditions' under which the Authorizer[1] trusts the Licensees[2] to perform an action.

[1] The Authorizer identifies the Principal issuing the assertion.

[2] The Licensees identifies the principals authorized by the assertion. More than one principal can be authorized, and authorization can be distributed across several principals through the

The exact semantics of the field is defined in RFC 2704. However parts of the language pertinent to our application are explained below.

Security attributes reside in the conditions section and are expressed in the form of logical statements. The conditions section's syntax is similar to that of a programming language "if statement". The section is usually broken into sub statements by using &&, ||, and parenthesis to construct logical conditions. For example the following phrase describes two security proposals supporting Telnet services (service_port= 23) using ESP with 3DES for encryption and finger services (service_port=79) using AH with SHA for authentication:

```
(local_filter_port == "23" &&
        esp_present == "yes" &&
        esp_enc_alg == "3des") ||
(local_filter_port == "79" &&
        ah_present == "yes" &&
        ah_auth_alg == "sha") -> "true";
```

Using the example in section 2(b) above, with security levels "high" and "low" and network modes "normal" and "impacted", the condition phrase is expanded (From [1]).

```
Conditions: ( (app_domain == "IPsec policy") && (
( (network_mode = "normal" &&
((security_level = "high" &&
 ((local_filter_port == "23" &&
        esp_present == "yes" &&
        esp_enc_alg == "3des") ||
(local_filter_port == "79" &&
        ah_present == "yes" &&
        ah_auth_alg == "sha"))) ||
((security_level = "low" &&
 ((local_filter_port == "23" &&
        esp_present == "yes" &&
        esp_enc_alg == "des") ||
(local_filter_port == "79" &&
        ah_present == "yes" &&
        ah_auth_alg == "des-mac")))) ||
(network_mode = "impacted" &&
((security_level = "high" &&
 ((local_filter_port == "23" &&
        esp_present == "yes" &&
        esp_enc_alg == "aes") ||
(local_filter_port == "79" &&
        ah_present == "yes" &&
        ah_auth_alg == "sha"))) ||
((security_level = "low" &&
 ((local_filter_port == "23" &&
        esp_present == "yes" &&
        esp_enc_alg == "3des") ||
(local_filter_port == "79" &&
        ah_present == "yes" &&
        ah_auth_alg == "sha-md5")))) -> "true";
```

use of `and' and threshold constructs.

As we notice the complexity of the language increases exponentially as we add more ports and parameters to it. The nesting of parenthesis to multiple levels makes writing a syntactically correct policy file almost impossible. In the following section, XML is analyzed to see if the technology could be used to make the task of specifying the Keynote policy file practical.

4 XML and Policy Representation

Extensible Markup Language (XML) [19] is a rapidly maturing technology with powerful real-world applications, particularly for the management, display and organization of data. XML is a technology concerned with the description and structuring of data. It is a subset of Standard Generalized Markup Language (SGML), with the same goals, but with much less complexity. XML is not a language but a standard for creating languages that meet the XML criteria. It describes a syntax that you use to create your own languages [7].

Data is separated from presentation in XML. XML structures the data, while style sheets format the data presentation . That makes it easier to use the data for multiple purposes. The same stylesheet can be used with multiple documents to create a similar appearance among them. Or alternatively multiple stylesheets can be applied to an XML document to provide different forms of presentation of the data. There are a variety of languages that can be used to create stylesheets such as Extensible Stylesheet Language Transformations (XSLT).

XML addresses the problem of data portability and software maintenance. Programmers have been structuring their data in an infinite variety of ways, and with every new way of structuring data comes much experimentation and testing to get it just right. If the data format changes, the methodologies to manipulate it also have to change, and the testing and tweaking has to begin again. The cycle of software maintenance will start all over again. With XML, there is a standardized way to structure the data and to extract the information we need. The extensibility of the language permits us to make changes as we need, without having to adjust the code that extracts information from the file.

4.1 XML DTDs and Schemas

The need to validate documents against a vocabulary led the creators of XML to include a method of checking validity in the XML recommendation. A document is valid if its XML content complies with a definition of allowable elements, attributes and other document pieces. By utilizing special 'Document Type Definition' syntaxes or DTDs, you can check the content of a document type with a special parser. The Document Type Definition (DTD) validation format has been used for many years to validate SGML and XML documents. As the use of XML and DTDs increased, some of the limitations of DTDs surfaced. Though these limitations restrict their use, DTDs are still useful for various applications.

However, with the release of XML schemas, a more powerful mechanism for validating XML documents is now available.

A Schema is the XML construct used to represent the data elements, attributes, and their relationships as defined in the data model. By definition, a DTD and a schema are very similar [20]. However, DTDs usually define simple, abstract text relationships, while schemas define more complex and concrete data and application relationships. A DTD doesn't use a hierarchical formation, while a schema uses a hierarchical structure to indicate relationships. XML Schema definitions are also commonly referred to as XSD.

4.2 XSLT

XML lets us structure our data in a hierarchical structure. This structure has some rigid rules and following them enables us to use other XML tools to access and manipulate the data without having to write code for it. However the structure may not suit an application and we may need an alternative representation of the data for either presentation purposes or for the purpose of manipulating it. Extensible Stylesheet Language Transformations, XSLT, is a language which can transform XML documents into any text-based format, XML or otherwise. It is a sub-component of a larger language called XSL [22]. XSL relies on finding parts of an XML document that match a series of predefined templates, and then applying transformation and formatting rules to each matched part. Thus once an XML document is created, XSLT can be used to transform the document into whatever other format we wish- HTML for display on web sites, a different XML-based structure for other applications, or even just regular text files.

Specifically, XSL is used to create stylesheets. An XSL engine uses these stylesheets to transform XML documents into other document types, and to format the output. Stylesheets define the layout of the output document and the location of the data in the source document. That is, "retrieve data from this place in the input document; make it look like this in the output". In XSL parlance, the input document is called the source tree, and the output document the result tree.

4.3 Advantages of XML for the Policy Specification Language

As described above we have a need to represent the intended IPsec policy in a form separate from the native KeyNote representation. Some of the advantages that would accrue by using XML are as follows:

4.3.1 Tools. Use of XML for specification of the KeyNote policy file lends itself to be used with the freely available, verified, tested and user-friendly tools. These tools include among others, XML editors, parsers, validators, translators etc. The availability of such tools and the extensive use of XML in modern communication protocols and other programs will enable users to manipulate XML files easily. Wide availability of such tools will also help in creating and maintaining the policy files over diverse systems without the need for an application specific editor.

4.3.2 Security. Interest in XML in recent years has resulted in huge investments in the field of XML security. The XML security features such as XML encryption and authentication will enhance the security of the policy file. This will also help, for instance, in selectively 'digitally signing' parts of the policy file. Thus a person signing a particular part of the policy file will only be responsible for the part he signed. Without the XML format it would be possible only to sign the entire file after for instance adding a part to it.

4.3.3 Platform Independence. It is possible to edit, maintain and distribute the XML policy file across different OS platforms.

4.3.4 Single Data Multiple Presentation. Once we represent the policy in an XML format it is possible to extract relevant information and present it in different forms that are more intuitive and useful to the administrator or the user. XSLT style sheets can be written and associated with the policy file to generate different presentation formats. Apart from presenting it in a more understandable and probably graphic format this will also help the administrator pin down any inaccuracies/inconsistencies/contradictions in the policy file. Intelligent agents can be written to audit the policy file and signal the administrator for errors in the policy file.

4.3.5 Consistency and Accuracy. XML Schemas and/or DTDs can be used to validate the XML file to see if it matches our specifications. Validating the policy file with a well-defined schema will enable errors to be picked up. This will trap all errors without having to go through the entire file manually. The use of generic schema generators and validators only makes this an easier task. This will also enable users to verify policy files received across the networks.

4.3.6 Extensible Format. An XML format will lend itself to extend the policy file to cater to new requirements in the policy file that come up in the future. Additional tags can be defined for elements and attributes as and when the need to incorporate them arises. This would not require changes to the application code as long as the structure of the document is maintained.

4.3.7 Ease of Use. The hierarchical nature of XML layout results in an easy to use and easy to manipulate format. It makes the file more modular and more easily understandable.

4.3.8 Semantic Content Use. The semantic content of the policy file enables future deployment of intelligent agents or roaming agents that can read policy files and report problems, and that can resolve conflicts between multiple systems by highlighting for instance the difference in the policies between them.

4.4 Integrating XML and Keynote Policy

The Keynote engine requires that the assertions, credentials and the policy files be specified in the syntax as specified in RFC 2704 and examined in section 'A' above. This structure restricts our ability to define any meaningful network security policy in an error free manner. Further, any policy file received in this format is not human readable, thus establishing a daunting requirement to verify its correctness and to detect security loopholes if any. Thus there is a clear problem of differentiation between data content and its representation. The same data is required by the Keynote engine in one format while on the other hand the format is not suited for human interpretation and validation. Specifying the policy data in an XML format enables us to use XSL to translate the data to any format needed, such as a more human readable form. Further specifying an XML Schema would provide us the benefit of validating the XML policy file for correctness prior to its transformation.

5 Design and Implementation

The first challenge to using XML for security policy specification was to determine an alternate representation of the policy logic in the form of an XML *user policy file*. The overall approach was to develop a format for the user policy file and then create a style sheet to transform it to the native KeyNote policy file format. This approach also provides the flexibility of later being able to extract useful administrative information from the user policy file.

Arriving at a format for the user policy file is a challenging task and there are multiple options available. The primary requirement is that the resulting XML file be well formed. During this research, multiple formats were considered. Each had its strengths and shortcomings. For instance one format would lend itself to an easy application design while another would permit more semantic content in the file format. The former therefore makes it easier to write an application such as a 'Policy Editor' while the latter results in a more descriptive self-defining file, which could be a good interchange format between multiple applications. However we realized that the specific format is not so significant as long as it has sufficient semantic content to be understandable. This results because the choice of element tag names, their sequence etc. is a personal preference: the power of XSL is always available for another user who wishes to use an alternative format. Thus arriving at a well annotated, self-defining and logical policy file format was the endeavor.

The XML User Policy file format we decided on is illustrated in the following example:

```
<Conditions>
  <ApplicationDomain app_domain="IPsec policy">
    <NetworkMode network_mode="normal">
      <SecurityLevel security_level="low">
          <Port local_filter_port="21"
                remote_filter_port="21">
            <Encapsulation esp_present="yes">
              <EncryptionAlgorithm esp_enc_alg="des" />
              <EncryptionAlgorithm esp_enc_alg="des3" />
            </Encapsulation>
          </Port>
          <Port local_filter_port="23"
                remote_filter_port="23">
            <Encapsulation esp_present="yes">
              <EncryptionAlgorithm esp_enc_alg="des3" />

              <EncryptionAlgorithm esp_enc_alg="aes" />
          </Encapsulation>
            <Ah ah_present="yes">
              <AuthenticationAlgorithm
                   ah_auth_alg="hmac-sha" />
              <AuthenticationAlgorithm
                   ah_auth_alg="des-mac" />
            </Ah>
          </Port>
        </SecurityLevel>
        <SecurityLevel security_level="medium">
          <Port local_filter_port="21"
                remote_filter_port="21">
            <Encapsulation esp_present="yes">
              <EncryptionAlgorithm esp_enc_alg="des3" />
              <EncryptionAlgorithm
                   esp_enc_alg="des-iv32" />
            </Encapsulation>
          </Port>
        </SecurityLevel>
        <SecurityLevel security_level="high" />
      </NetworkMode>
      <NetworkMode network_mode="impacted">
        <SecurityLevel security_level="low" />
        <SecurityLevel security_level="medium" />
        <SecurityLevel security_level="high" />
      </NetworkMode>
      <NetworkMode network_mode="crisis">
        <SecurityLevel security_level="low" />
        <SecurityLevel security_level="medium" />
        <SecurityLevel security_level="high" />
      </NetworkMode>
    </ApplicationDomain>
  </Conditions>
  <Dummy><![CDATA[
    ]]></Dummy>
</Policy>
```

Having arrived at the XML policy file format, XSL stylesheets are used to transform the policy file into the desired formats. Two stylesheets were designed using XSLT (Refer Figure 4).

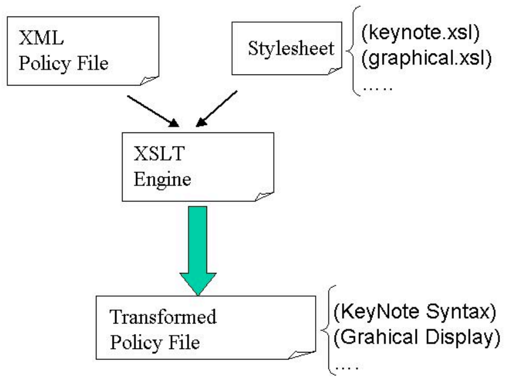


Figure 4. XSL Transformation of XML Policy Data in the User Policy File.

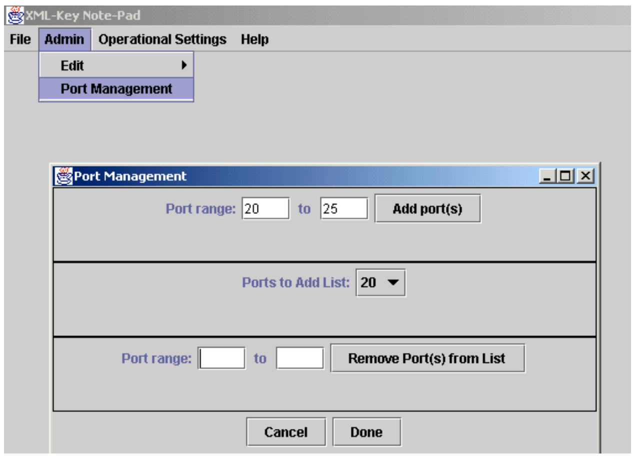
A stylesheet was created for transforming the file to the Keynote policy file format. An alternative stylesheet to transform the XML policy file to a more human readable, graphical web based format was also written. The transformed output of the XML Policy file using this template is shown in Figure 9.

5.1 Java based GUI

Though XML provides us with the flexibility to edit the policy file in any XML editor, it would still be convenient to provide a graphical user interface to manipulate the policy file. This would help in eliminating inadvertent errors and would also provide an automated entry into multiple elements without the need to edit each element content. This would enable global policy decisions to be applied throughout the policy file. An experienced system administrator could still capitalize on the use of the XML policy format and edit the file in the absence of the graphical user interface (GUI).

A Java-based GUI was therefore built to integrate various components of the software [10]. Drop down menus and dialog boxes guide the user to input various parameters required for the policy file. To enable maintenance of the GUI, called the *Policy-Editor*, a separate XML configuration file was used to feed the data for various drop down menus and combo/list boxes. This decoupling of the Java code from the configuration data will enable continued use of the Policy-Editor without the need to modify the Java code.

Figures 5 through 8 are screen shots of the Policy Editor. Figure 5 and Figure 6 select ports, and operational modes and security levels in the construction of a security policy. Figures 7 and 8 show the granular settings of encryption and authentication for particular ports. Figure 9 shows how the XSL transformation of the resulting policy file displays the policy in a graphical and more intuitive format.

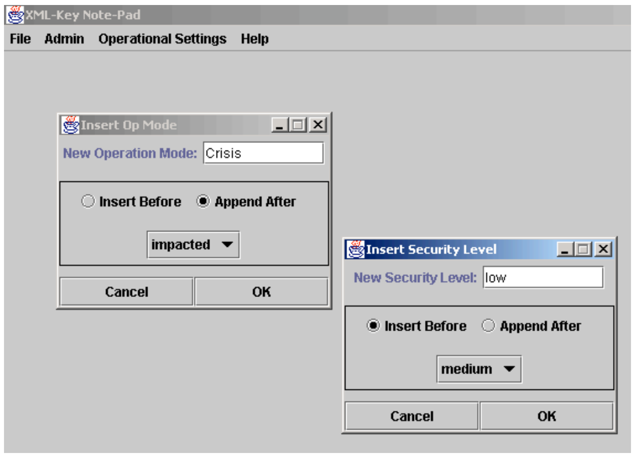


Port range: 20 to 25

Ports to Add List: 20

Port range: to

Figure 5. Managing Ports in the Admin Module.



New Operation Mode: Crisis

Insert Before [✓ Append After]

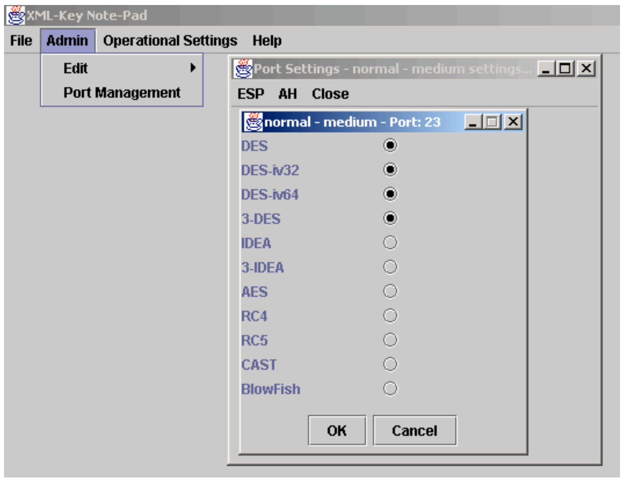
impacted

New Security Level: low

[✓ Insert Before] Append After

medium

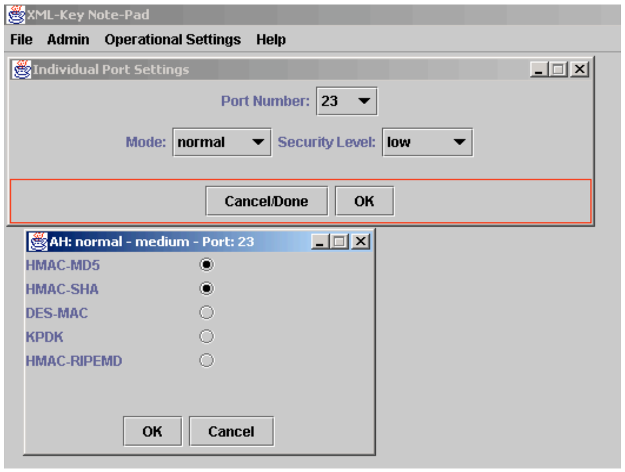
Figure 6. Admin Mode Settings for Security Level and Op Modes



normal - medium - Port: 23

DES	●
DES-iv32	●
DES-iv64	●
3-DES	●
IDEA	○
3-IDEA	○
AES	○
RC4	○
RC5	○
CAST	○
BlowFish	○

Figure 7. Encryption Settings For Individual Ports



Port Number: 23

Mode: normal Security Level: low

HMAC-MD5	●
HMAC-SHA	●
DES-MAC	○
KPDK	○
HMAC-RIPEMD	○

Figure 8. Authentication Settings for AH Mode Heading should be with figure

In addition to the polciy editor a generic XML editor, such as XML Spy (Copyright ©1998-2002 Altova GmbH) can be used to view and edit an XML Policy file. Figure 10 is a screen shot depicting the use of XML Spy editor to manipulate the XML Policy file directly. The result of schema validation can also be seen here. Figure 11 is the design view of the schema when viewed in XML Spy.

The security policy management toolkit is comprised of the Java based Policy-Editor and an XML editor such as XML Spy, XML Notepad, etc. The XML editors are not essential, but can aid in file manipulation, their transformation to multiple forms, and validation of schemas.

6 Future Work

Security policy management is a vast area and our work can be complemented with additional work in a number of areas to provide better tools for policy management. Listed below are several major items that will require attention.

6.1 Policy File Format

The XML policy file format currently specified could benefit from a more elaborate format with tags for other parameters. XML Namespaces and XML vocabularies could be utilized for a more comprehensive policy format [21]. Examples could involve incorporating other parameters such as algorithm key length, time-of-day parameters etc. The policy format should be able to accommodate other Boolean operators such as inequality definitions (<, >, !=) in the security policy management mechanism. For example, esp_enc_alg > DES could imply 3DES and AES if we have an ordering for the 'security strength' of each algorithm. Global policy statements such as encryption in crisis mode < 3DES, etc. should be possible. Inclusion of IP addresses in policy statements should also be made possible. These concepts have been demonstrated in the implementation. Addition of more parameters as stated above would however open up possibilities for inconsistencies in policy statements and the same will have to be carefully and formally worked out.

6.2 Schema Design and RELAX NG

W3C XML Schemas are complicated and hard to formulate. The schema generated in this work was automatically generated by XML Spy and modified manually to suit our current requirement. This schema language is very complex and permits us to define complex content models. The schema for instance could be made more specific or more general. This would depend on how we intend to formulate the policy statements. Future work could therefore focus on how a detailed policy encompassing various parameters such as IP addresses, encryption attributes etc could be specified. The interrelationship between different elements could also be specified, such as if 'Crisis' mode uses encryption then so should 'Normal' and 'Impacted' mode. In exploring more usage of Schema, alternate schema languages such as RELAX NG could be tried. This new language is gaining popularity due its simplicity and robustness. RELAX NG is the result of merging two popular schema languages: RELAX and TREX. The RELAX NG language is very similar to the W3C XML Schema language. RELAX NG has a much stronger foundation in mathematical models, which allows programmers to create highly optimized validation tools. In addition, RELAX NG omits many features that make W3C XML Schemas difficult to learn. RELAX NG, like W3C XML Schemas, is written in an XML Syntax and requires you to define the allowable elements and attributes within your instance documents. RELAX NG can be found on the Web at http://relaxng.org.

6.3 Policy Editor Enhancements

The policy editor interface, though complete and functional, can be improved upon. The particular improvements envisioned are as follows:

6.3.1 Data Binding. Data Binding is a concept where XML data can be read into applications as objects. By accessing the data as Java objects, manipulation becomes faster and managing large policy files would not slow down the system. Through the use of data binding the policy editor could be made more efficient and faster.

6.3.2 Global policy settings. The policy editor could be modified to enable global policy settings. For instance we could have a statement such as all ports should have a minimum encryption of DES or the maximum encryption algorithm for Crisis mode should not exceed 3DES etc. The global settings option could enter the default settings for all permissible ports and then more granular changes could be made.

6.3.3 Help. Help to guide the user to form syntactically correct policy statements and correct use of GUI could make the editor more complete. Context sensitive help could also be added.

6.3.4 Translation and viewing XML. XML translation and viewing currently need the help of any general purpose XML editor. Using Java packages such as Javax, the same could be incorporated into the GUI thus dispensing the need for XML editors for translation.

6.3.5 Schema validation. Validation of the XML document against DTD and Schema need to be incorporated into the GUI. The same is currently done using an XML tool such as XML Spy. DOM/JDOM/SAX could be used for the purpose.

6.3.6 Inconsistency and contradiction checks. As the policy file is extended to include global parameters and overlapping rules apply to a particular port or application, inconsistencies and contradictions would begin to emerge. The same would have to be considered and avoided. Various XML tools could help in achieving this. Distributed IPsec policy when considered would also give rise to multiple issues of policy consistencies.

6.3.7 Improvement in the look and feel of the user interface. The look and feel of the editor can always be improved to cater for user preferences and to avoid chances of introducing inadvertent errors. Context sensitive tool tips, toolbars and help could all be incorporated into the policy editor to give it a complete look.

6.4 XML Interface to Keynote.

It is felt that extending the XML policy language specified here to a broader XML specification and providing an XML processor in the Keynote engine itself would greatly enhance the use of Keynote. This would probably reduce the overhead of parsing in Keynote and provide the power of XML for better auditing and dynamic management of trust. XML security features could also be incorporated. For instance using XML signature, a user can sign for parts of the XML document i.e. a subset of the 'Elements'. Thus making him accountable for the parts signed by him only. By providing an XML interface to Keynote, application users could define their own versions of the policy language and use XSL for translating it into the desired Keynote format, which would be trivial, or alternatively they could use the vocabulary specified in the Keynote specifications.

7 Conclusion

Security policy management is a critical issue in the management of computer and networking resources. IPsec and Keynote provide a mechanism to implement a granular security policy. Previous research in the area of 'Quality of Security Service' demonstrates how an adaptive security policy can provide enhanced security with optimal utilization of network resources. The only missing link in the process is the difficulty in specifying a well-defined, granular, error free and consistent security policy in the language understood by the Keynote trust management engine. We have presented a solution to this problem in the form of an easy to use yet powerful security policy editor. The work demonstrates that use of XML technology as a middle layer provides us with a means to combine the security of Keynote with the simplicity of a policy editor. This novel approach also provides us all the benefits of XML, such as XSL and XML security. While XSL was extensively used, XML security tools could be used as follow up future work.

References

[1] Agar, C. *Dynamic Parameterization of IPsec,* Master of Science Thesis, Department of Computer Science, Naval Postgraduate School, December 2001.

[2] Keromytis, A. D., Ioannidis, J. and Smith, J. M., *Implementing IPsec*, In Proceedings of the IEEE *Global Internet (GlobeCom) 1997,* pp. 1948 - 1952. November 1997, Phoenix, AZ.

[3] Blaze, M., Feigenbaum, J., Ioannidis, J., and Keromytis, A. D, *The KeyNote Trust Management System Version 2, (RFC 2704,* Network Working Group, September 1999, http://www.ietf.org/rfc/rfc2404.txt

[4] Blaze, M., Feigenbaum, J., and Keromytis, A. D., *KeyNote: Trust Management for Public-Key Infrastructures,* In Proceedings of the *1998 Security Protocols International Workshop,* Springer LNCS vol. 1550, pp. 59 - 63. April 1998, Cambridge, England. Also *AT&T Technical Report 98.11.1.*

[5] Blaze, M., Ioannidis, J. and Keromytis, A. D. *Trust Management and Network Security Protocols*, In Proceedings of the *1999 Security Protocols International Workshop,* April 1999, Cambridge, England.

[6] Blaze, M., Ioannidis, J. and Keromytis, A. D., *Trust Management for IPsec,* In Proceedings of the Internet Society *Symposium on Network and Distributed Systems Security (SNDSS) 2001,* pp. 139 - 151. February 2001, San Diego, CA.

[7] Hunter, D., Cagle, K., Dix, C., Kovack,R., Pinnock, J., and Rafter, J., *Beginning XML 2nd Edition* , Wrox Press Ltd,2002.

[8] Doraswamy, N. and Harkins, D., *IPsec The New Security Standard for the Internet, Intranets, and the Virtual Private Networks*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1999.

[9] Irvine, C.E. and Levin, T., *Quality of Security Service, Proceedings of the New Security Paradigms Workshop*, Cork, Ireland, September 2000

[10] Java 2 Standard Edition, V1.2.2 API Specification, http://java.sun.com/products/jdk/1.2/docs/api/, Sun Microsystems, Inc., 1999.

[11] Kent, S and Atkinson, R, *Security Architecture* for the Internet Protocol, RFC2401, Network Working Group, November 1998, http://www.ietf.org/rfc/rfc2401.txt

[12] Leiseboer, J., *IPSEC – Security Architecture for IP, Part 2: Security Association*, http://www.chipcenter.com/eexpert/jleiseboer/jleiseboer036.

html, ChipCenter: The Web's Definitive Electronics Resource, Modified 12/05/2001.

[13] Maughan, D., Schertler, M., Schneider M., Turner J., *Internet Security Association and Key Management Protocol (ISAKMP), RFC 2408,* Network Working Group, November 1998, http://www.ietf.org/rfc/rfc2409.txt

[14] Mohan, R., XML Based Adaptive Policy Management in a Trust Management System Context, Masters Thesis, Naval Postgradute School, Monterey, CA, September 2002.

[15] Spyropoulou, E.., Agar, C. D., Levin, T., and Irvine, C., *IPsec Modulation for the Quality of Security Service,* NPS-CS-02-001, Naval Postgraduate School, Monterey, CA, January 2002.

[16] Spyropoulou, E.., Levin, T., and Irvine, C., *Demonstration of Quality of Security Service Awareness for IPsec,* NPS-CS-02-003, Naval Postgraduate School, January 2002.

[17] Thayer, R., Doraswamy,N., and Glenn, R., *IP Security Document Roadmap,* RFC 2411, Network Working Group, November 1998, http://www.ietf.org/rfc/rfc2411.txt

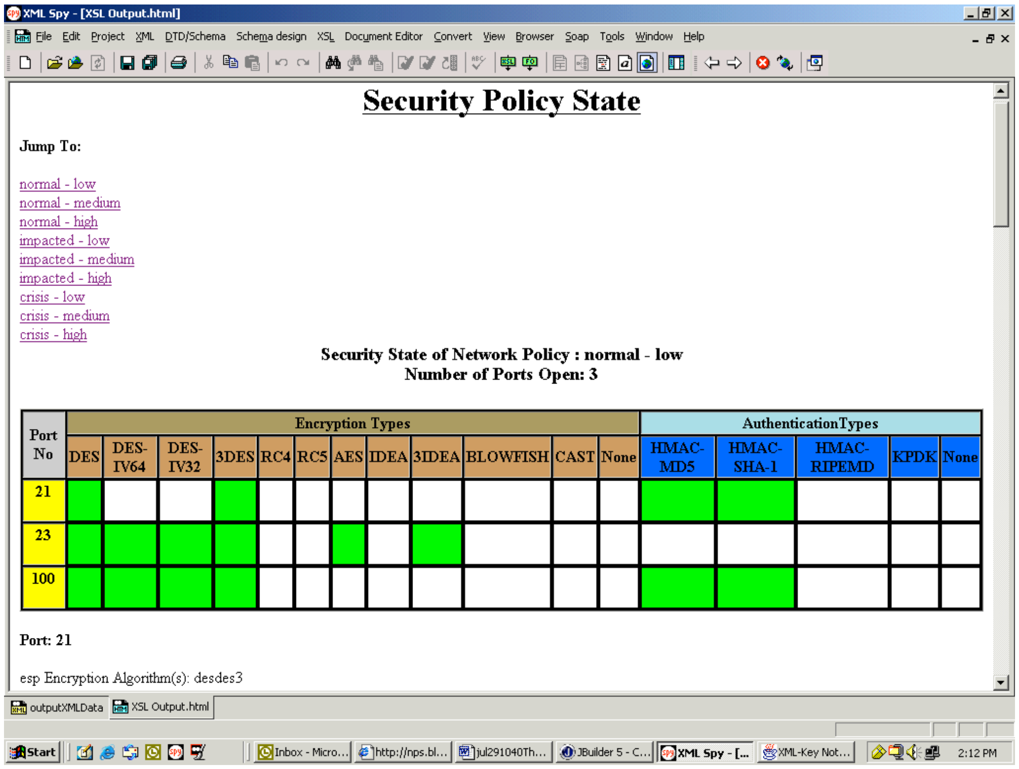
[18] Using IPsec *(Internet Security Protocol),* http://www.openbsd.org/faq/faq13.html, October 2001.

[19] XML Specification, http://www.w3.org/TR/2000/REC-xml-20001006 , Aug 2002

[20] XML Schema specifications, http://www.w3.org/TR/xmlschema-0, Aug 2002

[21] XML Namespace Recommendation, http://www.w3.org/TR/REC-xml-names/

[22] XSLT Specifications, http://www.w3.org/TR/xslt, Aug 2002 RFC2396


XML Spy - [XSL Output.html]
File Edit Project XML DTD/Schema Schema design XSL Document Editor Convert View Browser Soap Tools Window Help
outputXMLData
XSL Output.html
Start
Inbox - Micro...
http://nps.bl...
jul291040Th...
JBuilder 5 - C...
XML Spy - [...
XML-Key Not...
2:12 PM

Security Policy State

Jump To:

normal - low
normal - medium
normal - high
impacted - low
impacted - medium
impacted - high
crisis - low
crisis - medium
crisis - high

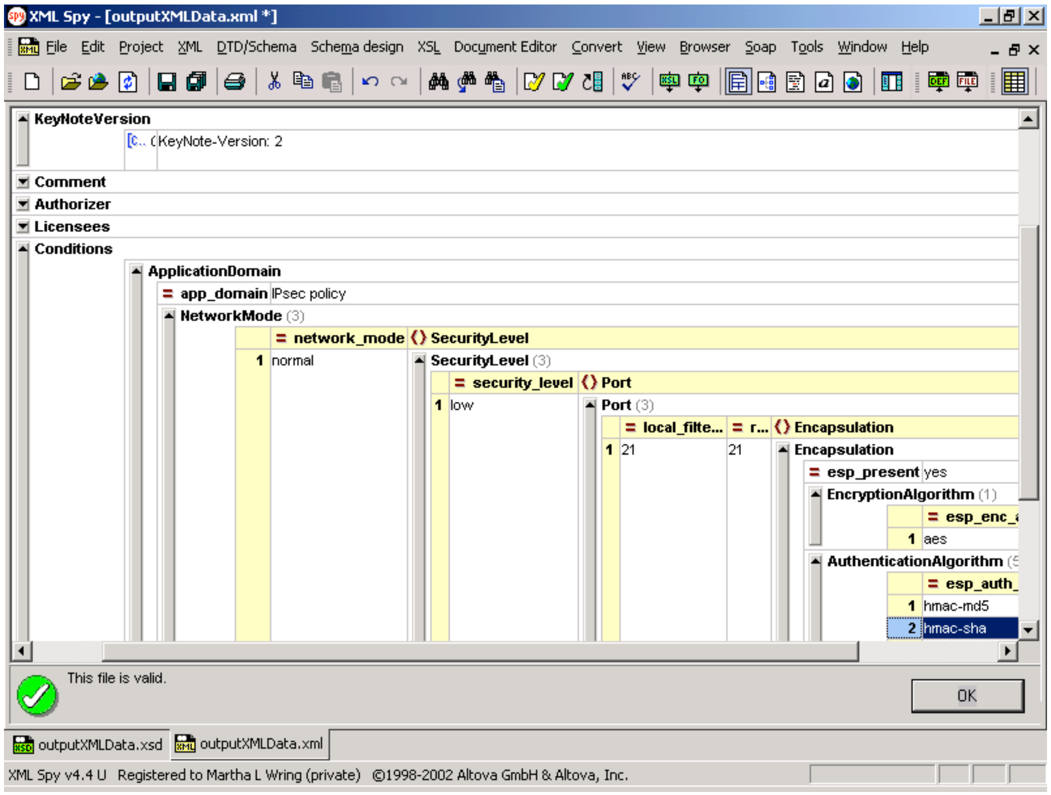
Security State of Network Policy : normal - low
Number of Ports Open: 3

Port No	Encryption Types												AuthenticationTypes				
	DES	DES-IV64	DES-IV32	3DES	RC4	RC5	AES	IDEA	3IDEA	BLOWFISH	CAST	None	HMAC-MD5	HMAC-SHA-1	HMAC-RIPEMD	KPDK	None
21	🟩			🟩									🟩	🟩			
23	🟩	🟩	🟩	🟩			🟩		🟩								
100	🟩	🟩	🟩	🟩									🟩	🟩			

Port: 21

esp Encryption Algorithm(s): desdes3

Figure 9. XSL Transformation of the Policy File

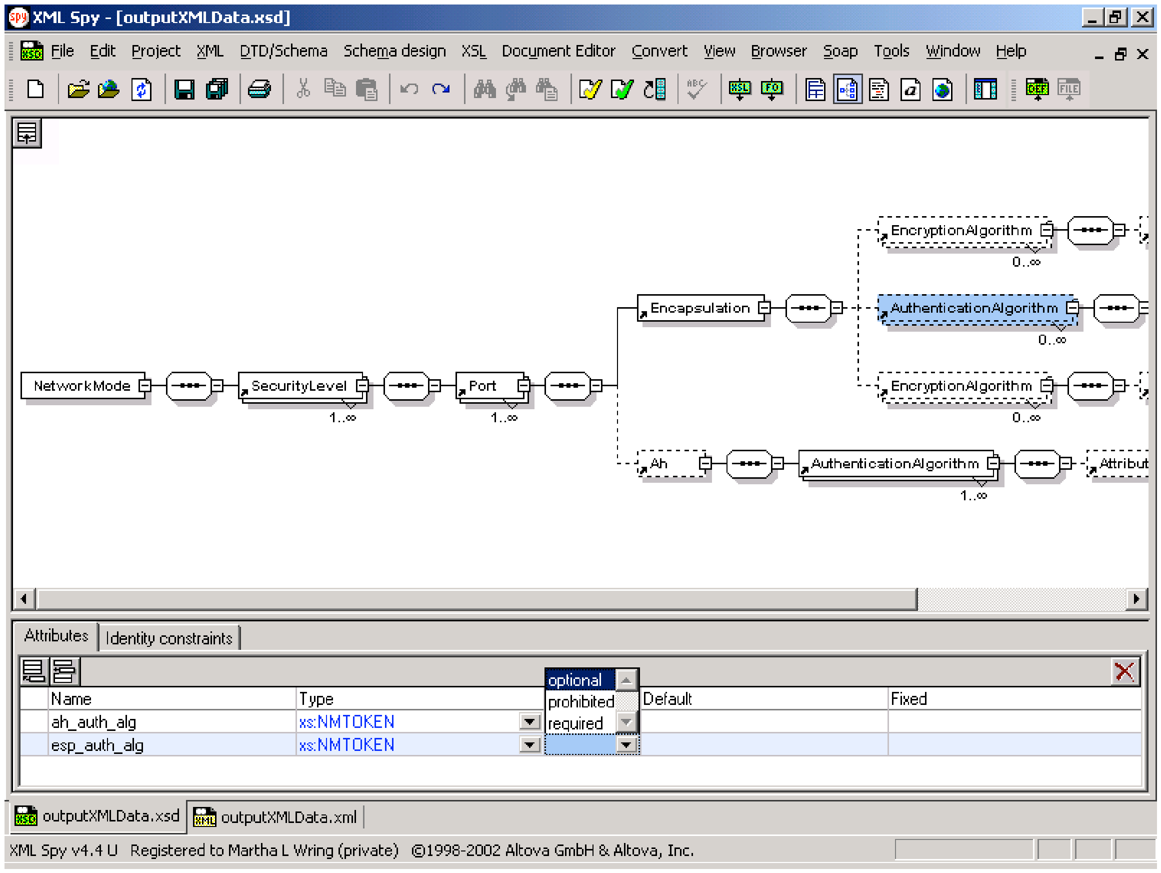

XML Spy - [outputXMLData.xml *]
File Edit Project XML DTD/Schema Schema design XSL Document Editor Convert View Browser Soap Tools Window Help
OK
outputXMLData.xsd
outputXMLData.xml

```
KeyNoteVersion
    KeyNote-Version: 2
Comment
Authorizer
Licensees
Conditions
    ApplicationDomain
        = app_domain IPsec policy
        NetworkMode (3)
            = network_mode   () SecurityLevel
            1 normal         SecurityLevel (3)
                                 = security_level   () Port
                                 1 low              Port (3)
                                                        = local_filte...  = r...  () Encapsulation
                                                        1 21              21      Encapsulation
                                                                                      = esp_present yes
                                                                                      EncryptionAlgorithm (1)
                                                                                          = esp_enc_
                                                                                          1 aes
                                                                                      AuthenticationAlgorithm (
                                                                                          = esp_auth_
                                                                                          1 hmac-md5
                                                                                          2 hmac-sha
```

This file is valid.

XML Spy v4.4 U   Registered to Martha L Wring (private)   ©1998-2002 Altova GmbH & Altova, Inc.

Figure 10. Editing and Validation of XML Policy File Using XML Sp



Name	Type	Use	Default	Fixed
ah_auth_alg	xs:NMTOKEN			
esp_auth_alg	xs:NMTOKEN			

Figure 11. Schema Design View of the XML Policy Document